Python Interface

The FTS Real Time System lets you manage your portfolio and create trading strategies using Python, giving you the opportunity to learn Python, obtain familiarity with machine learning techniques including open source libraries, and learn how to develop trading strategies.

- You create the strategy (or portfolio analytics) by writing a Python script
- The strategy can depend on your position and current prices
- It can also depend on historical data or any other data that you can access
- Your function returns a trade order
- The FTS Real Time Client (Windows version) runs your Python script and executes the trades
- You can run any version of Python you like, including Python libraries
- While we describe a how to implement a trading strategy, you can also use this to develop your own portfolio analytics to help manage your portfolio.

The operation is very simple. In the example below, we describe a simple trading strategy so you can learn the mechanics and get started quickly. You can download the sample script by clicking on:
http://www.ftsmodules.com/public/modules/ftsRT/Python/ConstantMixAnyPython.txt
(If you save it, you should rename it with a .py extension).

You can develop any trading strategy you like. Your Python script can download data, access packages, and so on. If you use Python packages, make sure that your Windows PATH is set correctly. See https://docs.python.org/3/using/windows.html for help on installing Python correctly and setting the PATH.

Example: A Constant Mix Strategy

As a simple illustration, consider the following "constant mix" strategy in the FTS 1000 Stock case:

- Monitor 3 tickers: IBM, CSCO, and INTC.
- Every 60 seconds, make sure that the dollar amount invested in each is (within rounding) equal to $100,000
    - This is called a "constant mix strategy"
    - The rounding occurs because the quantity we trade is an integer

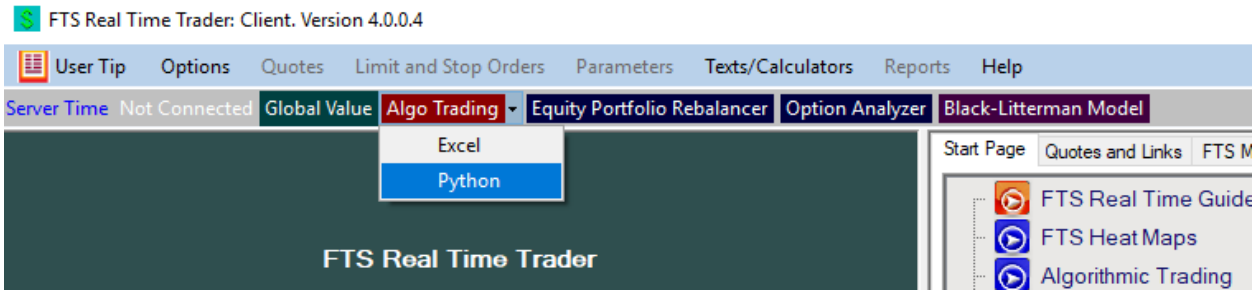
The Python script with the strategy is shown here, with comments on each section of the code. The comments (lines starting with #) are in green.

```
# You must import sys to be able to convert the arguments
# tickers must be specified as below as must the frequency (in seconds)
# Everything is case sensitive

import sys
tickers=['IBM','CSCO','MSFT']
frequency=60

# The next part converts the arguments into variables that are easier to use
# the arguments are passed as follows:
# argv[0] is the name of your python file

# then comes the data for the first ticker: last, q0,q2,...,q5
#       last=last traded price
#       q0=cash quantity
#       q1=margin quantity
#       q2=short quantity
#       q3=cash available
#       q4=VWAP of your position
#       q5=transaction cost per trade
```

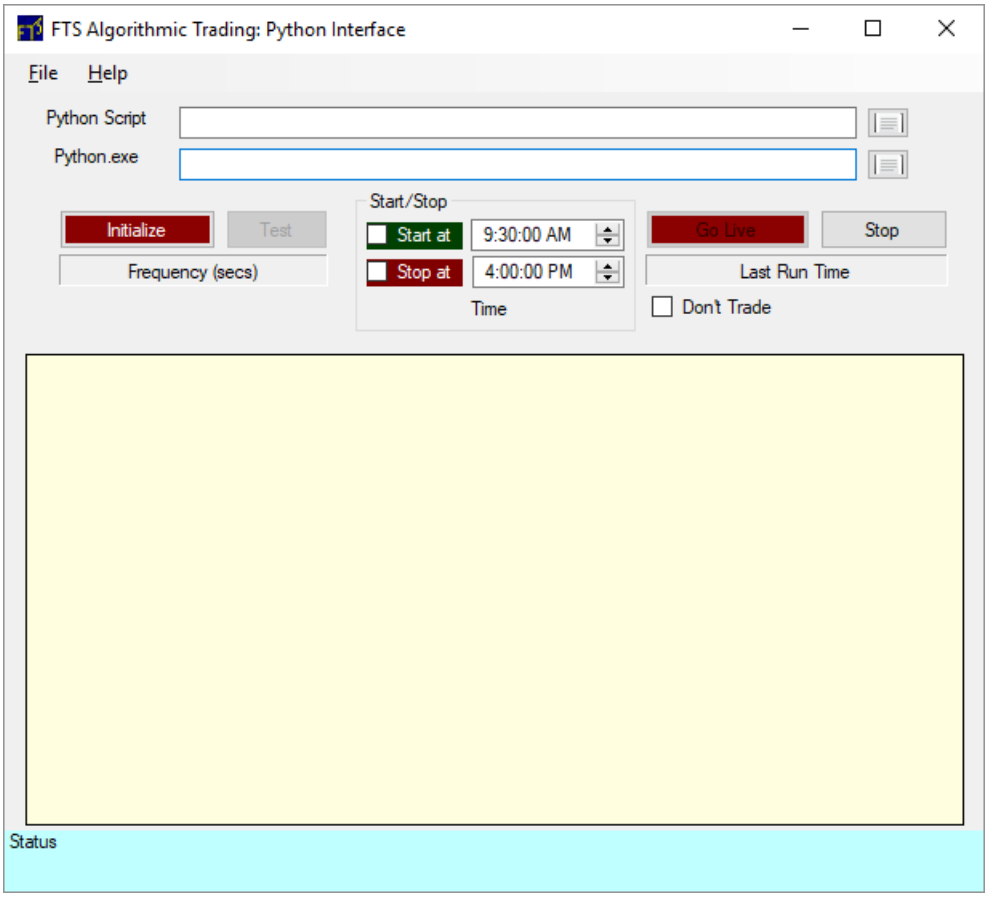
```
#           note that there are 6 elements of the quantity, numbered 0 to 5
# this is followed by the data for the second ticker
# the next part converts the arguments into last prices and position data by ticker
# the first is the number of arguments per ticker
nArg=int(sys.argv[1])
# in the future, we may pass more arguments per ticker; that is why we made this variable
i=1
last=dict()
qty=dict()
for tkr in tickers:
        i=i+1
        last[tkr]=float(sys.argv[i])
        qty[tkr]=list()
        for j in range(nArg):
                i=i+1
                qty[tkr].append(float(sys.argv[i]))

# the following declarations makes it easy to refer to the different parts of the qty:

cCashQty=0
cMarginQty=1
cShortQty=2
cCashAvaliable=3
cCVAP=4
cTransCost=5

# the trades are returned in a dictionary called returnString.
returnString=dict()
# each element of returnString is a key-value pair of the format returnString[ticker]=action/qty
# action is one of: cashbuy cashsell marginbuy marginsell shortsale shortcover
# qty is a number.
# Example: returnString['IBM']="cashbuy/10" means exactly what it says: buy 10 shares for cash.
# any order that is valid will be executed.
# So if you have cashbuy orders for multiple securities, they will all be executed at once.
# This allows you to trade baskets of securities.

# You can develop any trading strategy you like.  Your Python script can download data,
# and use any algorithm you like.
# IMPORTANT NOTE: if you use Python packages, make sure that your Windows PATH is set correctly.
# See https://docs.python.org/3/using/windows.html for help on installing Python correctly and setting the PATH.

# You can test your script without trading from the FTS Real Time Client.

# the next part shows how to create a constant mix strategy and return the trades you want to execute.
# It invests $100000 in each of the three tickers
# It does not check that you have enough cash available for the trades, you can add features like that if you like.

dollarAmount=100000.0
nTrade=0

for tkr in tickers:
        if last[tkr]>0:
                if qty[tkr][cCashQty]*last[tkr] < dollarAmount:
```

```
                    nTrade=dollarAmount / last[tkr] - qty[tkr][cCashQty]
                    returnString[tkr]='cashbuy/' + str(int(nTrade))
            elif qty[tkr][cCashQty]*last[tkr] > dollarAmount:
                    nTrade= qty[tkr][cCashQty] - dollarAmount / last[tkr]
                    returnString[tkr]='cashsell/' + str(int(nTrade))
    else:
            returnString[tkr]='blank'

# the next statement writes out the result and is read by the FTS Real Time Client and trades are executed

print(returnString);

# this should only be the ONLY print statement
```

Launch the FTS Real Time Client, and log in after selecting the FTS 1000 Stock Case (you must have a trading account set up for this case). At the top, you will see the Algorithmic Trading button, click on it and select "Python"



You will see:



Our position display shows us:

Stocks	Position	Last	Last Value
CISCO SYSTEMS (CSCO)	395	25.33	10,005.35
INTEL (INTC)	430	23.24	9,993.20
INTERNATIONAL BUSINESS MACHINE (IBM)	51	196.21	10,006.71

You can see that we have invested approximately $10000 in each stock. To get exactly $10000 in INTC, you would have to buy 430.2926 shares, we bought 430 shares.

Now, every 60 seconds, the FTS Real Time Client would run the macro again and trade if necessary as calculated by the trading algorithm.

That's it.

Extensions

Now, you know how to create trade orders by creating a *python script* called by the FTS Real Time Client with the specified arguments

You can now build any strategy you like:

- **Recent market data**: trades can depend on the bid, ask, and last prices.
- **Historical data**: you can access historical data from any source and use it within your python script to develop your trading strategy. Then, use the calculated results in the trade function.
- **Non-price data:** you can bring data on company fundamentals, even textual data that you interpret, such as news.

The example show you the basic steps: how to call the script, how to interpret data, and so on. Our strategy was simple so we could focus on showing you the basics, but you can probably figure out that you can create fairly complex strategies.

Example strategies

- **Contingent orders**: an order is executed only if one or more conditions are satisfied. Ultimately, any trading strategy results in a contingent order!
    - For example, you are interested in buying stock 1 or stock 2 but not both. So you specify a limit price on each in your function, and issue a buy order for the first stock when the limit price is reached.
- **Rebalancing**: like our constant mix example. Another example is re-allocating money across two different stock markets based on exchange rate movements.
- **Dynamic trading strategies and hedging**: buy or sell options or futures depending on stock price movements.
- **Basket trading**: buy a group of securities all at once when some conditions are met.
- **…….**

Summary and Caveats

- The point of the system is to help you learn python and understand how to develop and implement algorithmic trading strategies.
- This lets you go far beyond simple limit and stop orders and manual trade entry to the world of automated trading, so you can explore the world of quantitative strategies that play such an important role in today's markets
- This is an educational system, not designed as a high frequency trading system where you conduct a large number of trades every day. The idea is to focus on thinking about and implementing and testing strategies.
- You have to leave the real time client running and connected to our server for your algorithm to work
- If you start your algorithm, you should check periodically that you are connected to the server.

- You can lose the connection (in which case your strategy will stop) if there is an internet hiccup, if your computer (specially a laptop) goes to sleep, or, on very rare occasions, when we have to restart our server during the trading day.

You should test your strategy otherwise you could lose a bundle due to a coding error!